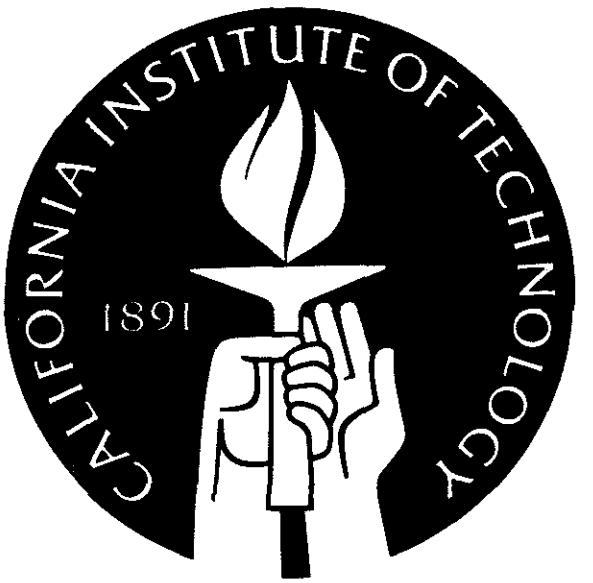

ASK is Transportable in Half a Dozen Ways

Bozena H. Thompson
and
Frederick B. Thompson

Computer Science Department
California Institute of Technology

5214:TR:86

ASK is Transportable in Half a Dozen Ways

Bozena H. Thompson
and
Frederick B. Thompson

5214:TR:86

ACM, Transactions on Office Information Systems
April 1985

ASK IS TRANSPORTABLE IN HALF A DOZEN WAYS

Bozena Henisz Thompson
Frederick B. Thompson

California Institute of Technology
Pasadena, California 91125

Abstract

This paper is an informal discussion of the technical issues and solutions encountered in making the ASK System transportable. A natural language system can be "transportable" in a number of ways. Although transportability to a new domain is most prominent, other ways are also important if the system is to have viability in the commercial marketplace.

"To transport" is a subcase of "to extend." To transport a system to a new domain is tantamount to starting with the system prior to adding any domain knowledge and extending it to incorporate the new domain. One may wish to add to a system that already has knowledge of one domain, the knowledge concerning a second domain, i.e., extend the system to cover this second domain. In the context of ASK, it has been natural to implement "extend", then achieve transportability as a special case.

We consider six ways in which the ASK System can be extended to include new capabilities:

- to a new domain
- to a new object type
- to access data from a foreign data base
- to a new natural language
- to a new programming language
- to a new computer family

I. What is ASK?

ASK, A Simple Knowledgeable System, is a total system for structuring, manipulation and communication of information. The primary ASK user interface is a simple dialect of English. ASK includes its own semantic net data base management system as well as many other capabilities for enhancing the user's ability to deal effectivly with his knowledge base. Although a number of papers have been published about ASK [1,2,3], we will start this paper,

whose principal topic is transportability of ASK, with a short series of examples of some of its English capabilities.

Here are some simple queries:

```
>What is the cargo of the Tokyo Maru?
 wheat
>How many tankers are there?
 2
>List the home port and destination of each tanker.
 tanker        home port destination
 British Star  London    Naples
 Christina     Piraeus   London
>What ships are longer than 450 feet?
 British Star
 Tokyo Maru
```

More complex queries illustrate such constructuions as relative clauses, negation and conjunction:

```
>What cities are the home ports of ships whose destination is London?
 Piraeus
 New York
>Are there any ships that do not have a cargo?
 yes
>What ships whose cargo is wheat have London or Oslo as destination?
          ships whose cargo is wheat
 London   Alamo
 Oslo     Karlgren
```

Ambiguity is handled in a straightforward way:

```
>What is the number of New York ships?
 There are 2 answers:
 (1) New York (destination) ships
 2
 (2) New York (home port) ships
 1
>Are there stewardesses of airlines whose age is greater than 50?
 There are 2 answers:
 (1) Are there (stewardesses of airlines) whose age is greater than 50?
 no
 (2) Are there stewardesses of (airlines whose age is greater than 50)?
 yes
```

Our study of user protocols (see [11]), have indicated that sentence fragments are a much-used style of communication. Echo, which we have found to be more ubiquitous in human face-to-face communication than most people realize, is used extensively in ASK English to clarify how the computer understands the user input:

```
>What is the destination of the British Star?
 Naples
>cargo?
 What is the cargo of the British Star?
 oil
>of the Orient Clipper?
 What is the cargo of the Orient Clipper?
 vehicles
>What about the Tokyo Maru?
 What is the cargo of the Tokyo Maru?
 wheat
```

Pronouns, which pose interesting linguistic problems, are handled in a rather sophisticated way:

```
>What is its destination?
 What is its [Tokyo Maru] destination?
 Yokohama
>Are there tankers whose home port is Piraeus?
 yes
>What are they?
 What are they [tankers whose home port is Piraeus]?
 Christina
```

Good diagnostics are an important part of a habitable system:

```
>What is the destination of the owner of Tokyo?
 There is no owner of Tokyo.
>Is the weight of the Kalgren coal?
 The individual coal cannot be the value of the
 number attribute weight.
>What is the square root of -5?
 You cannot take the square root of a negative number.
```

When the user input is not understood, a variety of corrective mechanisms are tried. Spelling correction is against the system's lexicon, rather than a pre-existing vocabulary, so that user-introduced vocabulary is covered:

```
>What is the crago of the Tokyo Mau?
 The following words are not in the vocabulary:
    crago  Mau
 Correction:
 What is the cargo of the Tokyo Maru?
 wheat
>Is coal the cargoof the Tokyo Maru.
 The following word is not in the vocabulary:
    cargoof
 Correction:
 Is coal the cargo of the Tokyo Maru?
 no
```

Users can change their data as easily as they can query it:

```
>What is the length and home port of the Tokyo Maru?
 length  home port
 foot
 520.2   Yokohama
>Home port of the Tokyo Maru is Hong Kong.
 Yokohama has been replaced by Hong Kong as home port of Tokyo Maru.
>Tokyo Maru's length is 564.5 feet.
 520.2 has been replaced by 564.5 as the length of Tokyo Maru.
>What is the length and home port of the Tokyo Maru?
 length  home port
 foot
 564.5   Hong Kong
```

Adding new vocabulary, including new attributes and classes, is straightforward:

```
>Create the class named person
 The class person has been added.
>Create a person named Capt. Ahab
 Capt. Ahab has been added as a member of the class person.
>attribute:captain
 The attribute captain has been added.
>The captain of the Karlgren is Capt. Ahab.
 Capt. Ahab has been added as the captain of the Karlgren.
>What person is the captain of each ship?
 ship       captain
 Karlgren   Capt. Ahab
```

Definitions are a common means of adding knowledge to the user interface:

```
>definition:Pearl of the Orient:Tokyo Maru
 Defined.
>definition:area:length times beam
 Defined.
>List the length, beam and area of the Pearl of the Orient.
 length   beam  area
 foot     foot  foot**2
  564.5     84    47418
>definition:super tanker:tanker whose length is greater than 1050
 Defined.
>definition:meter:39.37 * foot / 12
 Defined.
>What is each super tanker's area in square meters?
 super tanker  area
               meter**2
 British Star    17837.454
```

The only two verbs that are initially available are forms of "to be" and "to have," all the rest are added by paraphrase. Coordination, as illustrated by several of the queries in the following series, is a difficult problem which we have solved particularly smoothly:

```
>verb:ships "go" to Oslo:destination of ships is Oslo
 Defined.
>The British Star goes to what cities?
 London
>verb:ships "carry" coal to Oslo: there is a shipment whose carrier
 is ships, type is coal and destination is Oslo
 Defined.
>What are the types and destinations of shipments of the Continental?
 type    destination
 coal    Hong Kong
 wheat   Yokohama
>What does the Continental carry to Hong Kong?
 coal
>Does the Continental carry wheat to Hong Kong?
 no
>Does the Continental carry wheat and go to Hong Kong?
 carry wheat       yes
 go to Hong Kong   yes
>The Continental carries what to each city?
 Hong Kong coal
 Yokohama  wheat
>What city does the Continental carry coal to?
 Hong Kong
>What is carried to each city by the Continental?
 Hong Kong coal
 Yokohama  wheat
>What is carried to Hong Kong?
 coal
 vehicles
>What does the Continental carry?
 coal
 wheat
>What is carried?
 coal
 vehicles
 wheat
 oil
```

II. Transporting and Extending

These examples give some idea of what ASK does. We now turn to the subject of this conference as it applies to ASK, namely the transportability of ASK. As mentioned earlier, to transport" is a subcase of "to extend." For example, to transport a system to a new domain is tantamount to starting with the system prior to adding any domain knowledge and extending it to include knowledge of the new domain. One may wish to add to a system that already has knowledge of one domain the knowledge of a second domain, extending its knowledge base. In the context of ASK, it has been more natural to implement "extend," then acheive transportability as a special case.

A natural language system can be transportable in a number of ways. Although one way, transportable to a new domain, may be most prominent, other

ways are also important if the system is to have viability in the commercial market-place. In this paper we consider six ways in which ASK is extendable, thus transportable:

- to a new domain
- to a new object type
- to access data from a foreign data base
- to a new natural language
- to a new programming language
- to a new computer family

In this one paper, we will not be able to describe in any detail what we have done to make ASK extendable in each of these six ways. What we do want to do is to clarify the nature of the problems involved and the sense of our solutions.

III. Extending to a New Domain

You have seen in the examples of section I, how users can add new words and data to the knowledge base. Of course, if one wishes to create and maintain a small knowledge base, one can do so using these straight-forward methods. We do not construe that to be the problem of extending to a new domain. Rather, we envision the prior existence of a machine-readable data base, which will often be extensive. For example, if a natural language system were to be acquired by a company or agency, it is likely that they would have a number of data bases which they would eventually want to be part of their new system. It is the answer to that problem that we are seeking here.

In such cases, the existing management information system will have the means of outputting its information in formatted text files, since report generators are an essential and ubiquitous part of such systems. Thus ASK assumes that the data to be incorporated is in one or more text files. A tiny segment of such a data base might look like this:

```
NN   Christina      Piraeus
CC     oil            London
NN   Continental    Yokohama
CC     wheat          Yokohama
CC     coal           Hong Kong
NN   British Star   London
CC     oil            Naples
```

This particular data base consists of two kinds of physical records; first, one identified by "NN" that gives the name of a ship and its home port; followed by one or more "CC" records, each giving the cargo type and destination of a shipment carried by the ship. We will refer to this example to illustrate our methods.

Suppose the user wishes to add this data, and its associated vocabulary, to the ASK system. This can be done in three steps:

1) There will be certain words that will eventually be needed in the user's vocabulary and which will also be needed in instructing the system on how the data is to appear in the knowledge base. Usually they themselves will not appear in the data. Thus these words must be added first. In the example, this would be done as follows:

```
>classes: ship, shipment
>attributes:home port, destination, carrier, cargo type
```

2) The user must declare the formats of each physical record type and state how the contents of the various fields are to be built into the knowledge base. ASK has a system-guided dialogue, the Bulk Data Input Dialogue, for doing this. To initiate this dialogue, one simply types:

```
>bulk data input
```

The complete dialogue for the above example is given in Appendix A. The last question the computer asks in this dialogue is for the name of the text file containing the data.

3) The last step is for the system to open the given text file, read in the data, add the new words to its vocabulary, and build in the data. The user has nothing to do during this step except wait.

For a "typical" data base, steps 1 and 2 take a little planning; the structuring of a data base is still more art than science. All in all, if one knows the data base, the first two steps should not take more than an hour. The example Bulk Data Input dialogue, in Appendix A, should give you a feel for how long. The third step, for a data base of 10,000 or so items, is a matter of "cooking" over-night. The next morning, one can use the data, for example:

```
>List the cargo type and destination of the shipments of each ship.
ship          cargo type  destination
Christina     oil         London
Continental   wheat       Yokohama
              coal        Hong Kong

British Star  oil         Naples
```

IV. Extending to New Object Types

In a large percentage of installations where a system such as ASK might be used, there are extensive bodies of data and families of processing procedures that are not like typical "data bases." The management, e.g., indexing and retrieval, of these special files, on the other hand, is surely a data base problem. Further, since they have tangible connections to all sorts of aspects of the organization, they need to be tied into the organization's general knowledge base.

A typical instance is the graphic materials in a computer-aided design

environment. An engineering drawing is related to a part number, a designer, the various sub-assemblies containing the part, to texts such as specification lists. Thus it must exist as a data base item, retrievable using a variety of criteria. It also exists as a drawing per se. As a drawing, it may be processed by a variety of special procedures, for example one that produces from such a drawing a special file consisting of machining instructions. Note that such a procedure not only must have access to the internalized form of the drawing but also to other data; and its result is a new type of entity which itself constitutes an entry in the data base.

A ubiquitous case is correspondence and other text files. A letter can be considered from two points of view: one as a data base object -- where it is filed, who it has been sent to, cross references to other texts, to related people and to projects and offices to which it is relevant; the second, the text body itself. Such a text object relates to other objects in the data base through the regular data base procedures. Another family of procedures in the ASK System processes it in its text body role, namely procedures controlled by the word processor commands. A third family of procedures access this text object both in its text object role and its data base role; for example, the procedure that searches a class of texts to find those containing a given string, or the procedure that counts the words in a text object:

>Who wrote memos that contain the word "ICOT"?
>Display the items of my mail whose word count is less than 50.

The ASK System currently supports the following object types:

- individuals
- numbers
- matrices
- strings
- texts
- graphics
- images
- schedules

More to the point of this paper, ASK is so designed that it is easy to extend it to include altogether new object types. How does one do such an extension? Adding a new object type requires three steps:

- Add a simple declaration statement to the "object_type" module giving the name of the new type.

- Add a procedure, also in the "object_type" module, that returns an initialized form of an object of this type. For example, in the case of texts, such an initializing procedure asks the user to enter the text, puts it into the internal text structure and returns the address of this structure.

- Add new syntax rules covering the usages the end user is to employ in referring to objects of this new type.

- Add semantic procedures, corresponding to these syntax rules, that process these new objects.

Here is an example of such a syntax rule / semantic procedure pair which was added when we included image processing in ASK:

```
RULE
<noun_phrase:+image> => "enlargement of " <noun_phrase:+image>
                        " by " <noun_phrase:+number>
POST expand_proc
```

The corresponding procedure is:

```
procedure expand_proc;
   begin
(     (* Get address of image_data_set from first constituent *)
      ...
(     (* Get expansion_factor from second constituent *)
      ...
      call_C( EXPAN, image_data_set, expansion_factor );
      ...
(     (* Put resulting image_data_set address in appropriate
      ASK data struture *)
      end;
```

Once this and similar rules were added, the following sentence could be handled:

```
>Display the enlargement of Hong Kong by 10.
```

In this case of adding the "image" object type, a large family of image processing procedures was already available to us, namely the VICAR Image Processing System procedures of the Image Processing Laboratory of the Jet Propulsion Labortories. "EXPAN" is one of them, written in the programming language "C." We had prepared a Pascal callable procedure, "call_C," that will call such a C-procedure; you see its use above. Similar procedures, "call_FORTRAN," "call_COBOL,", "call_PROLOG,' etc. are also being added to ASK, as will be discussed in Section IV below.

V. Extending to Foreign Data Bases

Section III discussed how to extend ASK to a new knowledge domain. We consider three cases where that may not be a useful, or even a possible, thing to do. First, ASK may be put into a setting where large and important data bases already exist, together with a well-trained staff who routinely maintain these data bases and supply standard reports, often in hard copy, to key offices in the organization. It is unlikely that there will be a willingness on the part of management to shift the whole operation into a totally contained ASK implementation during the first months ASK is being used. Such a change usually takes a long time. Second, it may be that the data base is constantly and rather rapidly being updated and it would be too costly to repeatedly rebuild the ASK data base to keep up with these changes. Third, it

may be that access to the external data base is supplied by a commercial firm, is available only over the phone lines, and would be too expensive to acquire each working day.

In all three cases, it would be desirable to interface one's ASK System to the other external data bases. What one really wants is the capability to incorporate access to the external data base as an augmentation of the data resources already available. When one asks a question, it may require some of the data to be drawn from one external data base, other of the raw data may come from a second external data base, possibly over a phone line obtaining the very latest information, and finally the remainder of the data coming from one's own data base. The integration of the data from all three of these sources would be carried out and only the single, integrated result be presented to the user.

Suppose the computer is in the midst of responding to a user request and it needs to access data from a given foreign data base. Our solution to this problem is, at that moment, for the computer to configure itself so that it looks to the foreign data bases as a terminal, make the required request of the foreign data base which returns its respose as a text to this requesting "terminal." The user's computer then reconfigures itself back into ASK mode, processes the acquired text into its own internal data structures, and goes on.

How is the ASK system to get the requisite information needed to reconfigure itself to look like a terminal to a given foreign data base? A system-guided dialogue, i.e., an "expert system," has been designed and implemented that elicits this necessary information from the applications programmer serving the user (See [5]). This dialogue is initiated by typing:

>foreign access

The subsequent dialogue has two parts. In the first part, the system obtains the information that is of a hardware and system nature about the external data base system; for example, telephone number, baud rate, handshake convention, logon protocol, etc. In the second part of the dialogue, the user is asked for the access words he would like to use in referring to specific items of data he would like to access from the external data base, and for the translations of these individual words into the query language of the external data base system.

The first part of this dialogue can become very complicated, requiring knowledge of programming environments. Not only must the system understand how to present the request to the external data base system, but the computer must also understand the formats to be expected when the textual material is sent back to it; this includes such details as the page headers that may appear at intervals in the text, etc. In practice, the necessary information about most standard data base systems will have been prepared in advance, possibly by the ASK System supplier, so that the first part of the Foreign Access Dialogue can be bypassed by the user in extending his or her access to known types of external data bases (e.g., Datatrive, Sequel). This will, of course, greatly simplify the use of this ASK capability.

When these two aspects of the Foreign Access Dialogue are completed, the access path has been established. If the user now includes one or more of these access words in a query, the ASK System automatically opens the channel to the external data base, obtains the necessary data and integrates it into the context of the query.

It will be instructive to contrast these ASK methods with those of the Team Project at SRI International [6]. In each case, ASK and Team, we start with a user query. In Team:

- o the query is transformed into internal, interlinqua form, specifically into a form of the predicate calculus;
- o this internal form is translated into the query language of the external data base system, the translation embodying the entire logic of the original user input;
- o this query is passed to the external data base system and the response, as text, passed directly to the user.

In ASK:

- o the preprocessor, using the lexicon, identifies words marked as access words to external data bases; the methods of access and the specific queries to the external data bases, corresponding to these words, are available through the lexicon;
- o each of these corresponding queries is then passed to the specified external data base system which returns desired data for this single access word; similarly for the other access words;
- o the returned texts are put into internal ASK form, similar to the structure of other ASK data base objects;
- o ASK proceeds to process the query in its normal fashion.

In the case of a single query, ASK must transmit a great deal more raw data from the external data base than does Team. There is an ameliorating aspect however: if during a given session on the computer, a given access word is used, the transmission will take place only once. If ASK were considered just as a front-end to a single external data base, and forgo its own extensive data base back-end capability, which is the assumptions under which Team functions, ASK would clearly be inferior. However, ASK expects to hold most of the data it is using in its own data base and only go to one or more external data bases on occasion (otherwise the external data would have been brought into ASK using Bulk Data Input).

Team, on the other hand, is a great deal more dependent on the processing capability and efficiency of the external data base system. Dialogue, the associated Team "front end," is just that, a front end. Both the efficiencies

and complex logic of ASK in processing queries are not restricted by the external system. The importance of this becomes evident when one recalls that most data base systems in operational settings today are not very advanced nor efficient, including extant relational data base systems.

ASK supports access to several external data bases in response to a single query. It is conceivable that a logical form of a user query, such as Team uses, could be factored into several parts, together with a control logic to combine these parts. The parts could then be passed to each of several external data bases, the responses collected and combined, and a single answer given the user. This, however, is an almost intractable problem at this point in our technical knowledge.

In synopsis, Team provides an excellent natural language front end and a powerful way to relate that front end to an existent data base system. ASK, with its own back end, is able to access in a rather simplistic though effective way several external data bases and its own data base in response to a single user query. In both cases, Team and ASK, the method of establishing the correspondence with an external data base is a computer-guided dialogue. Each is quite complex, though very different in the information that the applications programmer must supply.

VI. Extending to Another Natural Language

Section I included a number of examples of ASK English. ASK French is also part of the ASK System [7]. Here is an example of ASK French:

```
>Quelle est la couleur du Baliet?
 rose
>Quelles sont les annees exceptionelles des vins roses?
 1928
 1966
 1976
>Est-ce-que le Pinot noir est un Champagne?
 non
>Quel est le prix des Bourgognes et des Champagnes?
             franc
 Bourgogne    31.05
              43.2
              72.
 Champagne   378.5
>definition:dollar: 8.6 francs
 Defini.
>Quel est le prix de chaque Bourgogne en dollar?
 Bourgogne                prix
                         dollar
 Nuits-Saint-Georges      5.023256
 Pouilly-Fuisse           8.372093
 Volnay                   3.610465
```

The first task of implementing a natural language in a computational linguistic system is to decide upon the data base structures and processing

utilities. Most current systems are designed as front ends to existing data base management systems and therefore these structures and processing utilities are given. Most of these systems assume that the "back ends" can be adequately characterized in terms of relational data bases, and on this assumption, translate the English input into a form of the predicate calculus -- as "interlingua" -- which can then be mapped into the query language of the back end system.

In the case of ASK English and ASK French, the data base structure is a semantic net with a rather general set of basic utility procedures for processing these nets. The syntactic structures were selected so that, on the one hand, they encompass the kinds of expressions a person would use in referencing such a data base, and on the other hand, their corresponding semantics can be expressed in terms of the given data structures and processing utilities. (Note that the interlingua step is completely by-passed.) From a linguist's point of view, the relationships that can be implemented in terms of current data base theory are gross, and put little strain on syntactic mechanisms for expressing data base relationships.

(It would be nice if there were a "universal" data base structure, but there is not. The "epistimological" levels (see [8], however also see [9]) of current data base theories leave little room for linguistic subtleties. To add at the syntactic level such a distinction as between mass and count nouns may add a semblance of sensitivity, but it is a hollow mask if the epistimological level of the data base does not reflect this distinction. Adding a semantic marker for such a distinction while giving it no other semantic significance only adds additional burden for the user in extending the vocabulary, since the value of this marker must be declared, without any compensating return in semantic power.)

Now that we have had the experience of adding a second natural language (that gross part of a natural language matching the semantic relationships expressable in terms of the data base structures), we see rather clearly the issues involved. There are two different aspects. The first, designing the syntax rules, is brought under control by the grossness of the semantic distinctions that need to be sensed by the system when it sees a user input. If, for example, we were to adopt the epistomological concept underlying the OWL System (Martin [9]), the task of syntax design would be a great deal more complex, because it would have to identify the syntactic clues that trigger the greater subtleties on the semantic side.

The second aspect is managing output to the user of the new language. ASK does not try to do any sentence generation. Referring back to the examples of section 1 and the above examples of ASK French, you will see that simple lists and tables account for most of the output. We expect graphical forms, e.g., pie charts and histograms, to be another useful form of output, but again requiring no sophisticated sentence generation. Thus the only real output problem was the generation of appropriate diagnostics. To illustrate, compare the following:

```
>Is the present king of France bald?
 There is no present king of France.
```

>Est-ce que l'actuel roi de France est chauve?
Il n'y a pas d'actuel roi de France.

Our solution is a simple one. The user can select the language to be used at any time. Subsequently, when the user has entered a query and in its processing an anomaly has been found, the procedure identifying the anamoly (and must therefore form the diagnostic output) can check a global variable to ascertain the language, and form the diagnostic accordingly.

The semantic procedures underlying ASK French are the same as those underlying ASK English. This opens the possibility that a given data base could be approached in either ASK English or ASK French, thus providing a true multi-lingual system. This capability is being implemented in ASK. The major design problem is the human interface mechanism for correlating the corresponding names in the two languages that should refer to the same data base item. For example, the lexicon entries for the English "wine" and the French "vin" must point to the self-same data base entry for the class of wines. The following illustrations will indicate our present plans for handling this problem:

>Create a class named wine
What is the French word for wine?

Responses:
(1) "vin(CR)" in which case additions are made to both lexicons using the same pointer to the added data base record for wines;
(2) "(CR)" in which case the same word is used in both French and English;
(3) "translate(CR)" in which case the same word "wine" is added to both lexicons, but in addition, the word is put on a list to be processed at a later time by an official translator.

We believe that ASK is transportable to other Natural Languages as well as French and English. ASK is the sister system to the USL System, developed at IBM Germany, Heidelberg [10]. Both ASK and USL have their roots in the REL System [11]. USL has both German, English and Spanish. From that experience, we feel that ASK German and ASK Spanish could be developed. A considerable part of the noun structures of Japanese was implemented in REL. On the basis of this and other work, it appears that an ASK Japanese would be possible.

VII. Extending to Other Programming Languages

We first must make it clear what programming languages have to do with ASK. To do so requires an introduction (we will make it as brief as possible, more complete documentation will be found elsewhere [4]) to the software engineering environment ASK provides for the applications programmer.

In ASK, the notion of a user's Context takes on a specific, formal meaning. An ASK Context consists of:

- ASK English (or French), including the math and statistical aspects, the text, graphics and image aspects, etc.

- o the special vocabulary and semantics of the user's domain, his or her data base including text and graphics files, definitions, and any extensions that have been made;

in sum, the complete conceptual environment for the user's interaction with the computer. An ASK station can hold many Contexts: one, say, for the administrative aspects of the user's concerns, others for each of the projects with which he or she is involved, a personal Context containing addresses and dates and personal correspondence, and perhaps others. These Contests can be interrelated in useful ways using the "basing operator" that can be invoked to base one Context on another (see Yu [13] for the concept of basing). All ASK Contexts are based, directly or indirectly, on the BASE Context; BASE contains ASK English and French, text processing, etc., but no application specific information.

Now to each ASK Context is associated a corresponding ASK Meta-Context. When a user Context, say AA, is based upon an existing Context, say BB, then a second Context, Meta-AA, is also created, based upon Meta-BB. The diagram of Figure 1 illustrates this.

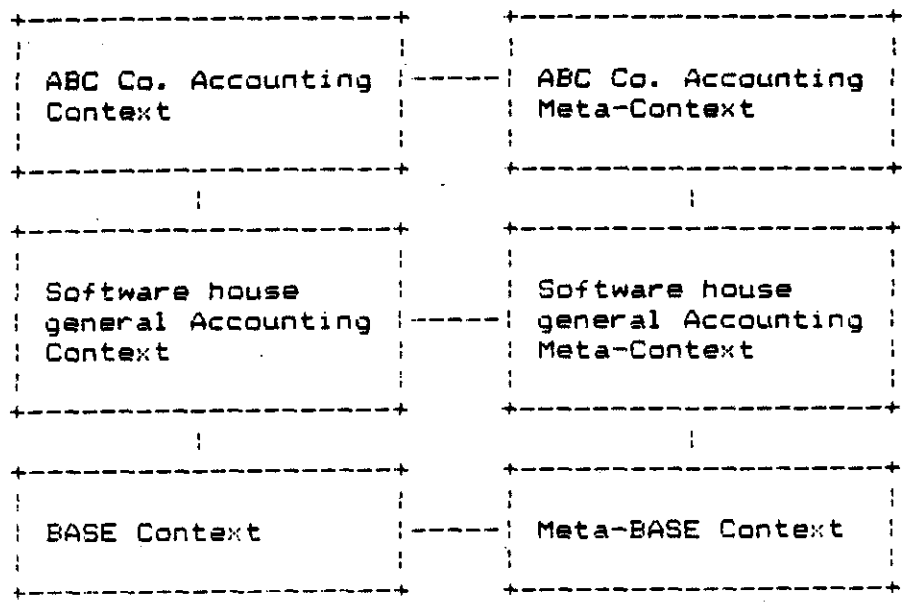

Figure 1: A Simple Context - Meta-Context Hierarchy

To extend a user's Context with a new capability, the application's programmer adds a new syntax rule and its associated semantic procedure. For example, suppose the applications programmer wishes to add a new numerical function which his user, say a banker, will refer to as "projected average return." This is a simple case, but one that will illustrate the process. The syntax rule to be added is:

```
RULE proj_avg_ret
<number attribute noun> => "projected average return"
proj_avg_ret_proc
```

where proj_avg_ret_proc is the associated semantic procedure. Once this rule and procedure have been added to the user's Context, the user can immediately ask:

>Display a bar graph of the projected average return of each of the high tech stocks of Mr. R. C. Rich.

How does the applications programmer add such a rule? The Meta-Contexts are the domain of the applications programmers. He enters the associated Meta-Context, then types "RULE," and the name he will use in referring to this rule. The ASK System metalanguage will then prompt him for the syntax and semantics:

```
>RULE proj_avg_ret
>Syntax: <number attribute noun> => "projected average return"
>Semantics: procedure proj_avg_ret_proc;
                begin
                ...
                end.
```

The Meta-Context knows to:

- collect the code for the semantic procedure
- add the necessary INCLUDES for system types and variables, utilities, etc.
- call the Pascal compiler
- link the resulting code into the ASK resident code
- put the syntax into the Context's grammar table with the proper pointer to the object code of the semantic procedure.

Just as the BASE Context contains ASK English, Meta-BASE contains Pascal. that is, a complete grammar for Pascal and the necessary semantics to carry out the above tasks. The same ASK language processor that handles ASK English also handles ASK Pascal, and thus all of the definition capabilites, spelling and other correction procedures, etc. are also available for processing the Meta-language statements. Of course, as one would imagine, MetaBASE also includes a wide range of debugging tools and Meta-Context extending capabilities. Thus, referring to Figure 1, the Software house furnishes an extensive set of utilities and meta-rules as part of Meta- Accounting that carry extensive knowledge of the accounting domain. The ABC Co.'s applications programmers are thus in a strong position to adapt the general accounting package to the practices of their particular company.

Now what about transportability? The user has ASK French as well as ASK English at his disposal; to go from one to the other, he simply states his choice. In parallel, we are adding the syntax for several programming

languages to MetaBASE so the applications programmer can also have a choice of programming languages when he is adding a new syntax rule and semantic procedure. Our present plans are to include Fortran, Cobol, C and Prolog. When a system like ASK is installed in an existing computer environment, this environment can be expected to include a large investment in both existing procedures and programmer expertise. The Meta-Context capability, including the choice of programming language, makes the protection of this investment a great deal easier.

Obtaining the grammar for a standard programming language is not particularly difficult. The semantics consists of two parts: collecting the code body, and presenting it to the appropriate compiler; the first of these is the same for all of the programming languages and already part of the metalanguage. The remaining task is to arrange for the semantic processor (the counter part of the LISP "eval" procedure) to recognize the particular programming language environment when calling a semantic procedure compiled in a particular programming language. In either a UNIX or an IBM-CMS operating system environment, this is easy for many of the standard programming languages, since such cross-language calls have already been facilitated. This implies, of course, that the right INCLUDEs were added to the source code at compile time, a step that we have accounted for above. The fact that these facilities exist in these environments can be exploited in introducing them in, say, a DOS environment.

When an applications programmer is required to maintain and extend a large, highly domain-dependent applications package, it is difficult for him, using current practices, to use anything but the programming language in which the package was originally programmed. By extending the MetaBASE Context to new programming languages, we are freeing the applications programmer to choose that programming language most natural for his immediate concerns.

VIII. Extending to Other Computer Families

Ninety-nine percent of ASK is written in standard Pascal. The machine-dependent parts of ASK are few in number, well-isolated and well-defined. It would seem that ASK should be easily transportable. Sadly, writing in standard Pascal does not make that true. It may be instructive to identify the kinds of problems we have encountered.

In the first place, the standard does not cover all aspects of a working Pascal. An example in point is string processing, concerning which the standard is particularly lax. We have implementated our own string procedures (using our own type: "ASK_string"), leaving as the only implementation-dependent part input and output translation of external strings to our own internal form. A second example is the conflict between ASCI and ECBDIC. Standard utilities translate between the two. However, to test whether an integer "ch" is the code for a letter, one is tempted to ask: "ch in ['a'..'Z']," using the ASCI assumption that the code runs successively from 'a', 'b', ..., 'z', 'A', ..., through 'Z'. But it doesn't in ECBDIC; thus this test must be expanded to: "ch in ['a'..'z'] or ch in ['A'..'Z']." Truly transportable Pascal is not by any means documented, and it is doubtful if it exists.

A second class of transportability problems arises because Pascals differ in word size. The sizes that need be considered are 16 bit, 32 bit and 64 bit machines. In ASK, there is a notion of "field type," ASK assumes a 64 bit field and thus basically a 64 bit real, 32 bit integer and a 64 bit pointer. However, we have isolated our type statements in a single submodule; which is included in all other modules. By selecting a particualr form of this module depending on the machine, we have been able to overcome this problem. For example, by switching this one module we can compile on IBM computers using either "PASCAL VS" or "PC PASCAL," including the handling in the latter case of the memory segmentation.

Outside the IBM family, we are approaching the transportability problem from another direction. We are working together with a software company, to be unnamed here, that has an excellent Pascal, and is preparing a series of compilers for that Pascal on many different computers. We mentioned above that there were a few machine-dependent aspects of ASK; these can all be subsumed under the machine-dependent aspects of their compilers, thus making ASK exactly as transportable as their Pascal. ASK is now running on the Sun Workstation using their compiler, and should be available for a wide range of computers via this route.

IX. Conclusion

An important aspect of any software that is intended to have an impact on the computing market-place is transportability, transportability in a number of different ways. Like other aspects of design, it does not materialize automatically, but needs to be included as a conscious consideration in all stages of design and implementation. The preceding sections have indicated the directions our work has taken to facilitate the transportability of the ASK System in six different ways.

Appendix: Bulk Data Input Dialogue for Example Data Base

```
>bulk data input
 You have entered the Bulk Data Input design dialogue.
      You can enter "help" at any time.
>What is the input data physical record length? 35
 Please describe the format of each physical record type in turn.

 Physical record type 1:
          1         2         3
 12345678901234567890123456789012345
>S.   N..... N.....
 Physical record type 2:
          1         2         3
 12345678901234567890123456789012345
>S.     N.....N.....
 Physical record type 3:
          1         2         3
 12345678901234567890123456789012345
>
```

```
 For each noun field, please enter its type, e.g. individual,
 number attribute, text class.
>Noun field <1,2>: individual
>Noun field <1,3>: individual
>Noun field <2,2>: individual
>Noun field <2,3>: individual

    Field:   Columns:  Length:  Type:
 Physical record #1:
    <1,1>    1-2       2        string
    <1,2>    6-18      13       individual noun
    <1,3>    20-32     13       individual noun
 Physical record #2:
    <2,1>    1-2       2        string
    <2,2>    8-21      14       individual noun
    <2,2>    22-35     14       individual noun
>Are these physical record formats correct? (y/n) y
 In what way are logical records distinguished from one another?
 B(lank record), S(pecial separator), L(ogical record identifier),
>P(hysical record identifier), F(ixed format): P
>Which field contains the physical record identifier? <1,1>
 What is the physical record identifier for the
>first physical record of a logical record? NN
>What is the physical record identifier for type 1: NN
>What is the physical record identifier for type 2: CC
 Do all physical records have exactly one physical record of each type,
 a fixed number of physical records of each type, or does the number of
>physical records in a logical record vary? O(ne), F(ixed), or V(aried): V
 Specify the actions to be taken if a blank field is encountered.
>Enter a field number: <1,2>
 What should be done if this field is blank? Respond with
 SF (skip field), SPR (skip physical record), SLR (skip logical record), or
 enter a substitute string for this field: SLR
>Enter a field name: <2,1>
 What should be done if this field is blank? Respond with
 SF (skip field), SPR (skip physical record), SLR (skip logical record), or
 enter a substitute string for this field: unknown
>Enter a field name:
 Specify the actions to be taken if a bad field is encountered.
>Enter a field number: <1,3>
 What should be done if this field is bad? Respond with
 SF (skip field), SPR (skip physical record), SLR (skip logical record), or
 enter a substitute string for this field: SF
>Enter a field name:
 Specify a field for which a translation table is needed.
>Enter a field name:
 Enter the conditions and actions to be taken upon completion of
 the processing of a physical record.
 For physical record of type 1:
>...Condition: <1,3> is a port?
>...Condition:
>...Action: <1,2> is a ship.
```

```
>...Action: Home port of <1,2> is <1,3>.
>...Action:
 For physical records of type 2:
>...Condition:
>...Action: Create a shipment.
>...Action: Its carrier is <1,2>.
>...Action: Its cargo type is <2,2>.
>...Action: Its destination is <2,3>.
>...Action:
 Enter the conditions and actions to be taken upon completion of
 the processing of a logical record.
>...Condition:
>...Action:
 If you would like to save this Bulk Data Input process for reuse then
>enter the name by which you will refer to it:
 The design of this BDI Process has been completed. It has not been saved.
>What is the name of the input data file?
```

References:

[1] B. H. Thompson and F. B. Thompson, "Introducing ASK: A Simple Knowledgeable System," Proceedings, Conference on Applied Natural Language Processing, ACL and NRL, Santa Monica, 1983, pp. 17-24.

[2] B. H. Thompson and F. B. Thompson, "ASK as Window to the World," Proceedings, International Conference on Systems, Man and Cybernetics, IEEE, Bombay and Delhi, 1984, pp. 1014-1018.

[3] B. H. Thompson and F. B. Thompson, "Customizing One's Own Interface Using English as Primary Language," Proceedings, South East Asia Regional Computer Conference, Hong Kong, 1984, pp. 15.1-15.16.

[4] B. H. Thompson and F. B. Thompson, "How to Get a Large Natural Language System into a PC," forth coming.

[5] A. C. Papachristidis, "Hetrogeneous Data Base Access," Ph. D. disertation, California Institute of Technology, 1984.

[6] B. J. Grosz, "TEAM, a Transportable Natural Language Interface System," Proceedings, Conference on Applied Natural Language Processing, ACL and NRL, Santa Monica, 1983, pp. 39-45.

[7] R. Sanouillet, "ASK French, a French Natural Language Syntax," Master's Thesis, California Institute of Technology, 1984.

[8] R. J. Brachman, On the Epistemological Status of Semantic Networks," Associative Networks, Editor: N. V. Findler, Academic Press, 1979, pp. 3-50.

[9] W. A. Martin, "Roles, Co-Descriptors and Formal Representation of Quantified English Expressions," Laboratory for Computer Science, MIT.

[10] M. Zoeppritz, "The Meaning of OF and HAVE in the USL System," American Journal of Computational Linguistics, Vol. 7, No. 2, 1981, pp. 109-119.

[11] B. H. Thompson and F. B. Thompson, "Shifting to a Higher Gear in a Natural Language System," Proceedings, 1981 National Computer Conference, Chicago, 1981.

[12] K. I. Yu, "Communicative Databases," Ph. D. disertation, California Institute of Technology, 1981.